

	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
1	BRS	L1	0	processing adj element SAME instruction adj buffer SAME (most adj often adj buffer) SAME profile	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/05/19 14:28	
2	BRS	L2	0	processing adj element SAME instruction adj buffer SAME (most adj often adj buffer)	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/05/19 14:28	
3	BRS	L4	2	3 and (most near3 often near3 buffer)	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/05/19 14:29	
4	BRS	L3	17	processing adj element SAME instruction adj buffer	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/05/19 15:16	
5	BRS	L5	60	(processing adj element or digital adj signal adj processor) SAME instruction adj buffer	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/05/19 15:17	
6	BRS	L6	43	5 not 3	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/05/19 15:17	
7	BRS	L8	0	6 and pointer and (predetermined near3 profile)	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/05/19 15:18	

	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
8	BRS	L7	16	6 and pointer and profile	USPAT; US-PGP UB; EPO; JPO; DERWE NT; IBM_TD B	2004/05/19 15:18	



US Patent & Trademark Office

[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☒ The ACM Digital Library ☐ The Guide

"instruction buffer" and decode and profile and pointer and first



THE ACM DIGITAL LIBRARY


[Feedback](#) [Report a problem](#) [Satisfaction survey](#)
Terms used **instruction**
buffer and **decode** and **profile** and **pointer** and **first** and **second** and **execution** and **predetermined** **profile** and **digital signal processor**

Found

29,741 of
132,857Sort results
by

relevance

Display
results

expanded form

[Save results to a Binder](#)[Search Tips](#)
☐ Open results in a new
window
[Try an Advanced Search](#)[Try this search in The ACM Guide](#)

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

Relevance scale ☐ ☐ ☐ ☐ ☐**1** [System-level power optimization: techniques and tools](#)

Luca Benini, Giovanni de Micheli

 April 2000 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**,
Volume 5 Issue 2

Full text available: pdf(385.22 KB)

 Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This tutorial surveys design methods for energy-efficient system-level design. We consider electronic systems consisting of a hardware platform and software layers. We consider the three major constituents of hardware that consume energy, namely computation, communication, and storage units, and we review methods of reducing their energy consumption. We also study models for analyzing the energy cost of software, and methods for energy-efficient software design and compilation. This survey ...

2 [Data and memory optimization techniques for embedded systems](#)
 P. R. Panda, F. Catthoor, N. D. Dutt, K. Danckaert, E. Brockmeyer, C. Kulkarni, A.
Vandercappelle, P. G. Kjeldsberg

 April 2001 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**,
Volume 6 Issue 2

Full text available: pdf(329.91 KB)

 Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

We present a survey of the state-of-the-art techniques used in performing data and memory-related optimizations in embedded systems. The optimizations are targeted directly or indirectly at the memory subsystem, and impact one or more out of three important cost metrics: area, performance, and power dissipation of the resulting implementation. We first examine architecture-independent optimizations in the form of code transformations. We next cover a broad spectrum of optimizations ...

Keywords: DRAM, SRAM, address generation, allocation, architecture exploration, code transformation, data cache, data optimization, high-level synthesis, memory architecture customization, memory power dissipation, register file, size estimation, survey

3 [The effect of reconfigurable units in superscalar processors](#)

Jorge E. Carrillo, Paul Chow

 February 2001 **Proceedings of the 2001 ACM/SIGDA ninth international symposium on
Field programmable gate arrays**

Full text available: pdf(208.11 KB)

 Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper describes OneChip, a third generation reconfigurable processor architecture that integrates a Reconfigurable Functional Unit (RFU) into a superscalar Reduced Instruction Set Computer (RISC) processor's pipeline. The architecture allows dynamic scheduling and dynamic reconfiguration. It also provides support for pre-loading configurations and for Least Recently Used (LRU) configuration management. To evaluate the performance of the OneChip architecture, several off-the-shelf ...

Keywords: OneChip, reconfigurable processors, superscalar processors

4 Trace-driven memory simulation: a survey

Richard A. Uhlig, Trevor N. Mudge

June 1997 **ACM Computing Surveys (CSUR)**, Volume 29 Issue 2

Full text available:  pdf(636.11 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

As the gap between processor and memory speeds continues to widen, methods for evaluating memory system designs before they are implemented in hardware are becoming increasingly important. One such method, trace-driven memory simulation, has been the subject of intense interest among researchers and has, as a result, enjoyed rapid development and substantial improvements during the past decade. This article surveys and analyzes these developments by establishing criteria for evaluating trace-driven ...

Keywords: TLBs, caches, memory management, memory simulation, trace-driven simulation

5 Reconciling responsiveness with performance in pure object-oriented languages

Urs Hölzle, David Ungar

July 1996 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 18 Issue 4

Full text available:  pdf(537.19 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)


Dynamically dispatched calls often limit the performance of object-oriented programs, since object-oriented programming encourages factoring code into small, reusable units, thereby increasing the frequency of these expensive operations. Frequent calls not only slow down execution with the dispatch overhead per se, but more importantly they hinder optimization by limiting the range and effectiveness of standard global optimizations. In particular, dynamically dispatched calls prevent standard ...

Keywords: adaptive optimization, pause clustering, profile-based optimization, run-time compilation, type feedback

6 Reducing the branch penalty by rearranging instructions in a double-width memory

Manolis Katevenis, Nestoras Tzartzanis

April 1991 **Proceedings of the fourth international conference on Architectural support for programming languages and operating systems**, Volume 19, 25, 26 Issue 2, Special Issue, 4

Full text available:  pdf(1.39 MB)

Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

7 A text-compression-based method for code size minimization in embedded systems

Stan Liao, Srinivas Devadas, Kurt Keutzer

January 1999 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, Volume 4 Issue 1

Full text available:  pdf(184.16 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)



We address the problem of code-size minimization in VLSI systems with embedded DSP processors. Reducing code size reduces the production cost of embedded systems we use data-compression methods to develop code-size minimization strategies. In our framework, the compressed program consists of a skeleton and a dictionary. We show that the dictionary can be computed by solving a set-covering problem derived from the original program. To execute the compressed code, we describe two methods ...

Keywords: code size optimization, compression

8 Energy-effective issue logic

Daniele Folegnani, Antonio González

May 2001 **ACM SIGARCH Computer Architecture News , Proceedings of the 28th annual international symposium on Computer architecture**, Volume 29 Issue 2

Full text available:  pdf(774.80 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)
 [Publisher Site](#)

The issue logic of a dynamically-scheduled superscalar processor is a complex mechanism devoted to start the execution of multiple instructions every cycle. Due to its complexity, it is responsible for a significant percentage of the energy consumed by a microprocessor. The energy consumption of the issue logic depends on several architectural parameters, the instruction issue queue size being one of the most important. In this paper we present a technique to reduce the energy consumption ...

Keywords: adaptive hardware, energy consumption, issue logic, low power

9 EXPLORER: a retargetable and visualization-based trace-driven simulator for superscalar processors

Trung A. Diep, John P. Shen, Mike Phillip


December 1993 **Proceedings of the 26th annual international symposium on Microarchitecture**

Full text available:  pdf(1.56 MB) Additional Information: [full citation](#), [references](#), [citations](#)

10 Transient fault detection via simultaneous multithreading

Steven K. Reinhardt, Shubendu S. Mukherjee

May 2000 **ACM SIGARCH Computer Architecture News , Proceedings of the 27th annual international symposium on Computer architecture**, Volume 28 Issue 2


Full text available:  pdf(151.56 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Smaller feature sizes, reduced voltage levels, higher transistor counts, and reduced noise margins make future generations of microprocessors increasingly prone to transient hardware faults. Most commercial fault-tolerant computers use fully replicated hardware components to detect microprocessor faults. The components are lockstepped (cycle-by-cycle synchronized) to ensure that, in each cycle, they perform the same operation on the same inputs, producing the same outputs in the absence of faults ...

11 Techniques for efficient inline tracing on a shared-memory multiprocessor

S. J. Eggers, David R. Keppel, Eric J. Koldinger, Henry M. Levy

April 1990 **ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 1990 ACM SIGMETRICS conference on Measurement and modeling of computer systems**, Volume 18 Issue 1

Full text available:  pdf(1.12 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)


While much current research concerns multiprocessor design, few traces of parallel programs are available for analyzing the effect of design trade-offs. Existing trace collection methods have serious drawbacks: trap-driven methods often slow down program execution

by more than 1000 times, significantly perturbing program behavior; microcode modification is faster, but the technique is neither general nor portable. This paper describes a new tool, called MPTRACE, for collecting tr ...

12 Performance of Lisp systems

Richard P. Gabriel, Larry M. Masinter

August 1982 **Proceedings of the 1982 ACM symposium on LISP and functional programming**

Full text available:  pdf(1.45 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper describes the issues involved in evaluating the performance of Lisp systems. We explore the various levels at which quantitative statements can be made about the performance of a Lisp system, giving examples from existing implementations wherever possible. Our thesis is that benchmarking is most effective when performed in conjunction with an analysis of the underlying Lisp implementation and computer architecture. We examine some simple benchmarks which have been used to measure ...

13 Retargetable tools for embedded software: Instruction set compiled simulation: a technique for fast and flexible instruction set simulation

Mehrdad Reshadi, Prabhat Mishra, Nikil Dutt

June 2003 **Proceedings of the 40th conference on Design automation**

Full text available:  pdf(196.91 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)


Instruction set simulators are critical tools for the exploration and validation of new programmable architectures. Due to increasing complexity of the architectures and time-to-market pressure, performance is the most important feature of an instruction-set simulator. Interpretive simulators are flexible but slow, whereas compiled simulators deliver speed at the cost of flexibility. This paper presents a novel technique for generation of fast instruction set simulators that combines the benefit ...

Keywords: compiled simulation, instruction abstraction, instruction set architectures, interpretive simulation

14 Regular contributions: DSP architectures: past, present and futures

Edwin J. Tan, Wendi B. Heinzelman

June 2003 **ACM SIGARCH Computer Architecture News**, Volume 31 Issue 3

Full text available:  pdf(1.27 MB)

Additional Information: [full citation](#), [abstract](#), [references](#)

As far as the future of communication is concerned, we have seen that there is great demand for audio and video data to complement text. Digital signal processing (DSP) is the science that enables traditionally analog audio and video signals to be processed digitally for transmission, storage, reproduction and manipulation. In this paper, we will explain the various DSP architectures and its silicon implementation. We will also discuss the state-of-the-art and examine the issues pertaining to pe ...

15 Speculative precomputation: long-range prefetching of delinquent loads

Jamison D. Collins, Hong Wang, Dean M. Tullsen, Christopher Hughes, Yong-Fong Lee, Dan Lavery, John P. Shen

May 2001 **ACM SIGARCH Computer Architecture News , Proceedings of the 28th annual international symposium on Computer architecture**, Volume 29 Issue 2

Full text available:  pdf(995.50 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

 [Publisher Site](#)


This paper explores Speculative Precomputation, a technique that uses idle thread context in a multithreaded architecture to improve performance of single-threaded applications. It attacks program stalls from data cache misses by pre-computing future memory accesses in available thread contexts, and prefetching these data. This technique is evaluated by simulating the performance of a research processor based on the Itanium™ ISA supporting

Simultaneous Multithreading. Two primary for ...

16 Bytecode compression via profiled grammar rewriting

William S. Evans, Christopher W. Fraser

May 2001 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2001 conference on Programming language design and implementation**, Volume 36 Issue 5

Full text available:  pdf(1.03 MB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

This paper describes the design and implementation of a method for producing compact, bytecoded instruction sets and interpreters for them. It accepts a grammar for programs written using a simple bytecoded stack-based instruction set, as well as a training set of sample programs. The system transforms the grammar, creating an expanded grammar that represents the same language as the original grammar, but permits a shorter derivation of the sample programs and others like them. A program's de ...

Keywords: bytecode interpretation, context-free grammars, program compression, variable-to-fixed length codes

17 Slice-processors: an implementation of operation-based prediction

Andreas Moshovos, Dionisios N. Pnevmatikatos, Amirali Baniasadi

June 2001 **Proceedings of the 15th international conference on Supercomputing**

Full text available:  pdf(236.51 KB)


Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

We describe the Slice Processor micro-architecture that implements a generalized operation-based prefetching mechanism. Operation-based prefetchers predict the series of operations, or the computation slice that can be used to calculate forthcoming memory references. This is in contrast to outcome-based predictors that exploit regularities in the (address) outcome stream. Slice processors are a generalization of existing operation-based prefetching mechanisms such as stream buffers where the ...

18 Control flow optimization for supercomputer scalar processing

Pohua P. Chang, Wen-mei W. Hwu

June 1986 **Proceedings of the 3rd international conference on Supercomputing**

Full text available:  pdf(1.04 MB)


Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Control intensive scalar programs pose a very different challenge to highly pipelined supercomputers than vectorizable numeric applications. Function call/return and branch instructions disrupt the flow of instructions through the pipeline, degrading the utilization of the pipelined datapaths. This paper describes control flow optimization for scalar processing using an optimizing compiler. To obtain program control flow information, a system independent profiler has been integrated into th ...

19 Compiler-directed early load-address generation

Ben-Chung Cheng, Daniel A. Connors, Wen-mei W. Hwu

November 1998 **Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture**

Full text available:  pdf(1.88 MB)

Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

20 Improving Java performance using hardware translation

Ramesh Radhakrishnan, Ravi Bhargava, Lizy K. John

June 2001 **Proceedings of the 15th international conference on Supercomputing**

Full text available:  pdf(254.91 KB)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)





State of the art Java Virtual Machines with Just-In-Time (JIT) compilers make use of advanced compiler techniques, run-time profiling and adaptive compilation to improve performance. However, these techniques for alleviating performance bottlenecks are more effective in long running workloads, such as server applications. Short running Java programs, or client workloads, spend a large fraction of their execution time in compilation instead of useful execution when run using JIT compilers. In ...

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2004 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)